Gebze Technical University

Programming Languages

CSE 341

Homework 5

Report

Elif Akgün

1801042251

1. Implementation

I implemented some functions for this project. I explained all of them one by one below.

1. start

At the beginning of the program, it calls this function and program begins. This function reads input file line by line. At each line, it decides the line is *fact*, or *rule*, or *query* with checkParentheses function. If the line is fact, it calls fact function; if the line is rule, it calls rule function. If the line is query, it calls query function <u>and it writes T if the query is true, or it writes empty list (()) if query is false, and it writes the value list if it is exist</u>. This function reads lines until end of the file.

This function does not return anything.

2. checkParentheses

This function decides the read line is fact, query or rule. For this purpose it uses parenthesses. If the clause is a fact, the body will be nil, so list ends with empty list. If the clause is a query, head will be nil, so list begins with empty list.

This function returns 0 if the clause is fact, returns 1 if the clause is rule, and returns 2 if the clause is query.

3. fact

If the clause is fact, this function is called. It converts the read line from string to list. Then it appends this list to *fact-list* list.

This function does not return anything.

4. rule

If the clause is rule, this function is called. It converts the read line from string to list. Then it appends this list to the *rule-list* list.

This function does not return anything.

5. query

If the clause is query, this function is called. It converts the read line from string to list. Then it returns this list.

6. solve-query

If the clause is query, this funciton is called to solve the query. First of all, it checks facts with *check-facts* function. If the query equals one of facts, it returns true. If the query hits some facts, it returns these facts' values. For example, if there are parent(ali, veli) and parent(ayşe, veli), then the query is ?- parent(X,veli); it returns (“ali” “ayşe”). If the query does not hit facts, then it checks rules with *check-rules* function. If there is such a rule and the query is true, it returns true, otherwise it returns nil.

7. check-facts

First of all, it checks all facts and it compares all facts with query. If one of them equal with query, it returns true. If there is no such a fact, it checks query’s variables and facts again. If one of them hits it takes values then returns them. (Like parent example above.) Fort his purpose, it uses *check-facts-helper* function.

8. check-facts-helper

If one of fact hits with query, this function completes variables with values, then returns this values. Fort his purpose it uses unification method.

9. check-rules

It checks the body of rule recursively. If one of them is nil, it returns nil; otherwise it returns true. It checks all rules. If the query’s name and one of rules’ name is same, then if checks size of query and this rule. If both of them equal, it check the query to find whether there is variable. If query include at least one variable (like ?- father(ali,Y).), then it calls *solve-rule* method to find suitable values.

10. solve-rule

First of all, this function checks query’s variables. If some of them are known, it takes them to variables and creates new query. For exampe, let the rule is mother(X,Y) :- parent(X,Y), female(X). Then let the query is ?- mother(X,ali). We know Y is ali. Actually this is unification. Then we check parent(X,ali) and female(X) facts.

This funstion also uses resolution method to solve queries. For example, let the rule is mother(X,Y) :- parent(X,Y), female(X). Then let the query is ?- mother(X,Y). In this situation, with using resolution method, we also check parent(X,Y), female(X) facts. This is resolution.

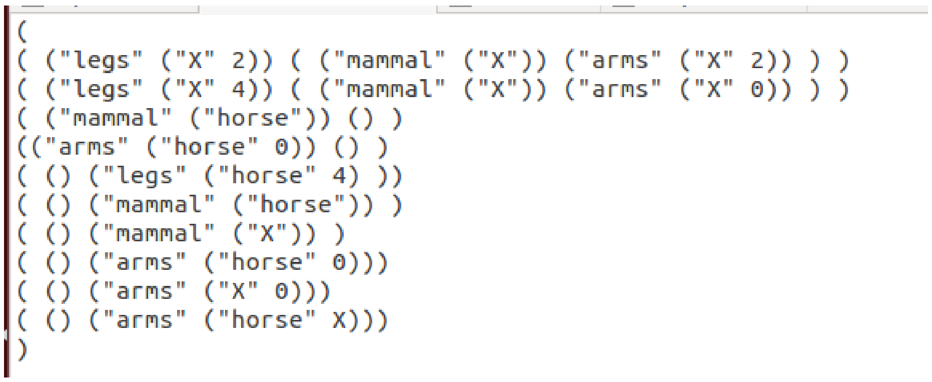
11. check-list

This function takes a list and an item as parameters. It checks whether this item exist in the list. If it exist, it returns true, otherwise it returns nil.

2. Tests

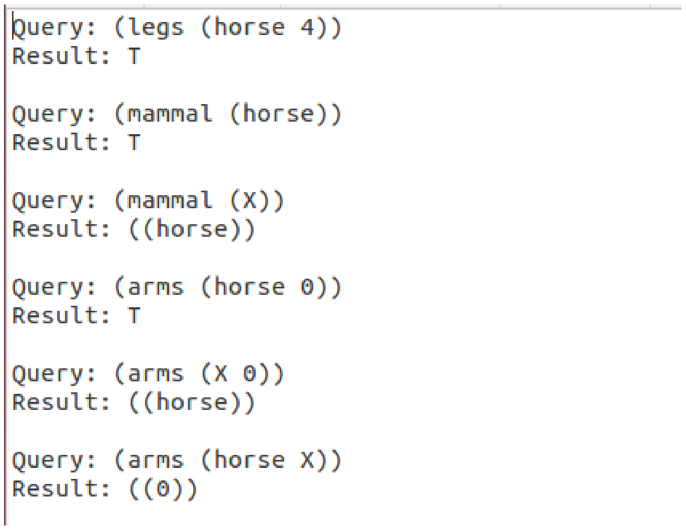
Input file’s fotmat must be like below examples to program run correctly. First line must be ‘(‘ and last line must be ‘)’ because I parsed file accordingly.

1. Sample Input/Output 1



```
(
( ("legs" ("X" 2)) ( ("mammal" ("X")) ("arms" ("X" 2)) ) )
( ("legs" ("X" 4)) ( ("mammal" ("X")) ("arms" ("X" 0)) ) )
( ("mammal" ("horse")) () )
(("arms" ("horse" 0)) () )
( () ("legs" ("horse" 4) ))
( () ("mammal" ("horse")) )
( () ("mammal" ("X")) )
( () ("arms" ("horse" 0)))
( () ("arms" ("X" 0)))
( () ("arms" ("horse" X)))
)
```

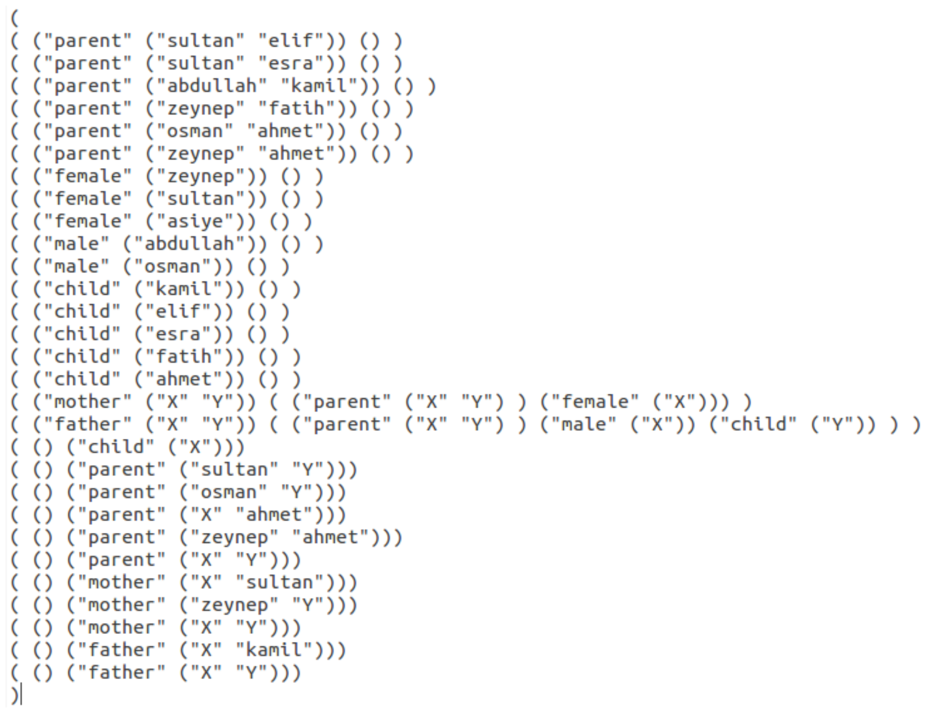
Figure 1: input.txt



```
Query: (legs (horse 4))
Result: T

Query: (mammal (horse))
Result: T

Query: (mammal (X))
Result: ((horse))

Query: (arms (horse 0))
Result: T

Query: (arms (X 0))
Result: ((horse))

Query: (arms (horse X))
Result: ((0))
```

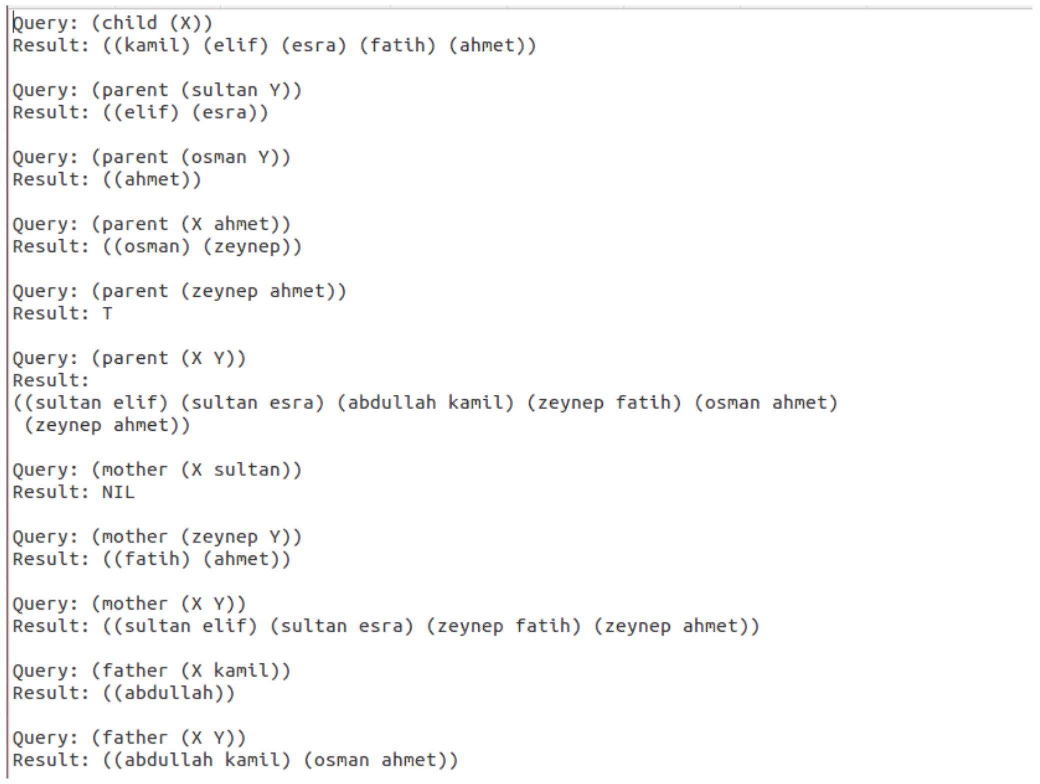
Figure 2: output.txt

2. Sample Input/Output 2



```
(
( ("parent" ("sultan" "elif")) () )
( ("parent" ("sultan" "esra")) () )
( ("parent" ("abdullah" "kamil")) () )
( ("parent" ("zeynep" "fatih")) () )
( ("parent" ("osman" "ahmet")) () )
( ("parent" ("zeynep" "ahmet")) () )
( ("female" ("zeynep")) () )
( ("female" ("sultan")) () )
( ("female" ("asiye")) () )
( ("male" ("abdullah")) () )
( ("male" ("osman")) () )
( ("child" ("kamil")) () )
( ("child" ("elif")) () )
( ("child" ("esra")) () )
( ("child" ("fatih")) () )
( ("child" ("ahmet")) () )
( ("mother" ("X" "Y")) ( ("parent" ("X" "Y") ) ("female" ("X"))) )
( ("father" ("X" "Y")) ( ("parent" ("X" "Y") ) ("male" ("X")) ("child" ("Y")) ) )
( () ("child" ("X")))
( () ("parent" ("sultan" "Y")))
( () ("parent" ("osman" "Y")))
( () ("parent" ("X" "ahmet")))
( () ("parent" ("zeynep" "ahmet")))
( () ("parent" ("X" "Y")))
( () ("mother" ("X" "sultan")))
( () ("mother" ("zeynep" "Y")))
( () ("mother" ("X" "Y")))
( () ("father" ("X" "kamil")))
( () ("father" ("X" "Y")))
)
```

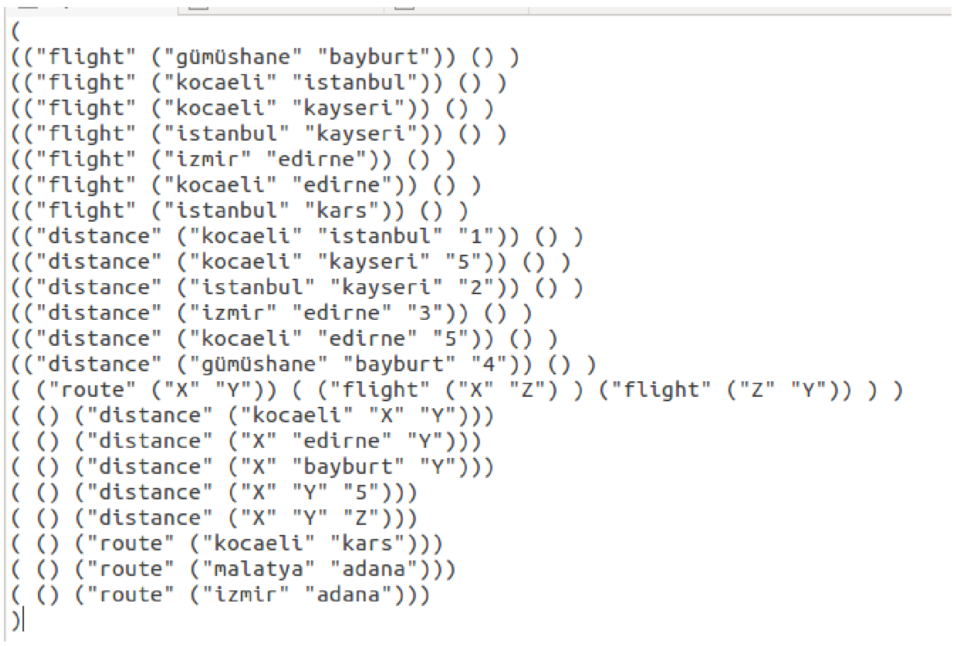
Figure 3: input.txt



```
Query: (child (X))
Result: ((kamil) (elif) (esra) (fatih) (ahmet))

Query: (parent (sultan Y))
Result: ((elif) (esra))

Query: (parent (osman Y))
Result: ((ahmet))

Query: (parent (X ahmet))
Result: ((osman) (zeynep))

Query: (parent (zeynep ahmet))
Result: T

Query: (parent (X Y))
Result:
((sultan elif) (sultan esra) (abdullah kamil) (zeynep fatih) (osman ahmet)
 (zeynep ahmet))

Query: (mother (X sultan))
Result: NIL

Query: (mother (zeynep Y))
Result: ((fatih) (ahmet))

Query: (mother (X Y))
Result: ((sultan elif) (sultan esra) (zeynep fatih) (zeynep ahmet))

Query: (father (X kamil))
Result: ((abdullah))

Query: (father (X Y))
Result: ((abdullah kamil) (osman ahmet))
```

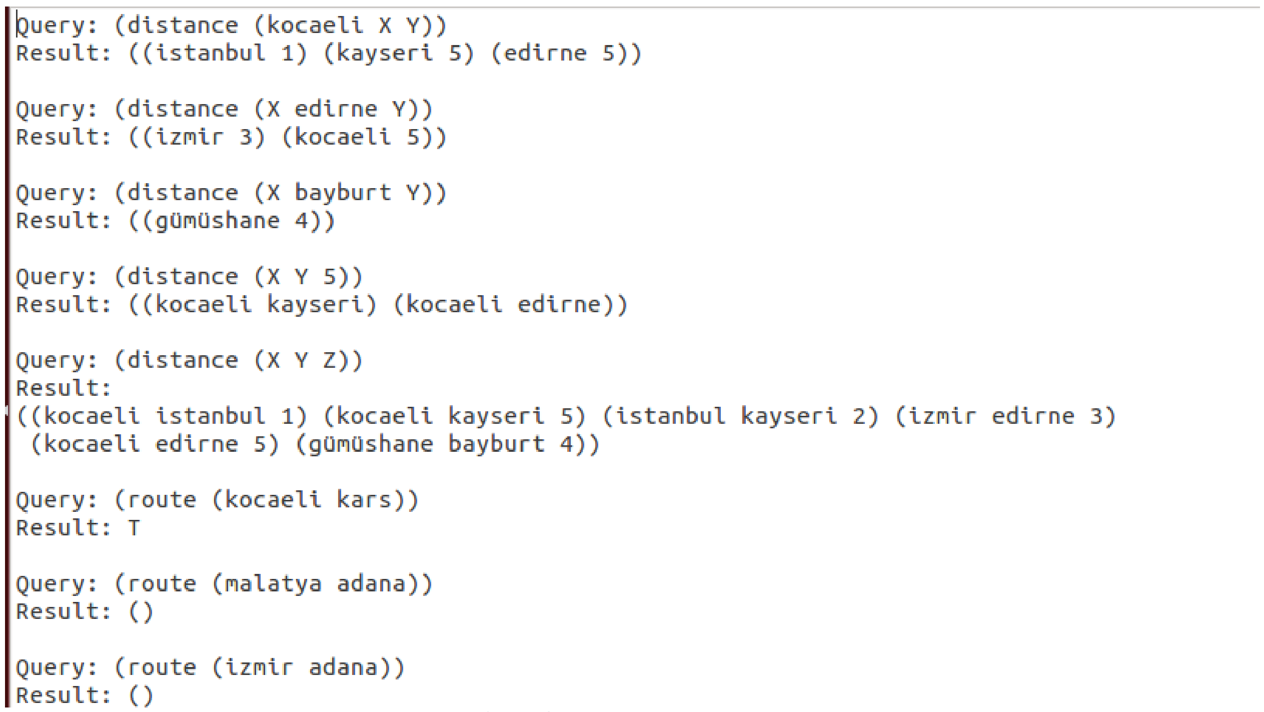
Figure 4: output.txt

3. Sample Input/Output 3



```
(
(("flight" ("gümüshane" "bayburt")) () )
(("flight" ("kocaeli" "istanbul")) () )
(("flight" ("kocaeli" "kayseri")) () )
(("flight" ("istanbul" "kayseri")) () )
(("flight" ("izmir" "edirne")) () )
(("flight" ("kocaeli" "edirne")) () )
(("flight" ("istanbul" "kars")) () )
(("distance" ("kocaeli" "istanbul" "1")) () )
(("distance" ("kocaeli" "kayseri" "5")) () )
(("distance" ("istanbul" "kayseri" "2")) () )
(("distance" ("izmir" "edirne" "3")) () )
(("distance" ("kocaeli" "edirne" "5")) () )
(("distance" ("gümüshane" "bayburt" "4")) () )
( ("route" ("X" "Y")) ( ("flight" ("X" "Z") ) ("flight" ("Z" "Y")) ) )
( () ("distance" ("kocaeli" "X" "Y")))
( () ("distance" ("X" "edirne" "Y")))
( () ("distance" ("X" "bayburt" "Y")))
( () ("distance" ("X" "Y" "5")))
( () ("distance" ("X" "Y" "Z")))
( () ("route" ("kocaeli" "kars")))
( () ("route" ("malatya" "adana")))
( () ("route" ("izmir" "adana")))
)
```

Figure 5: input.txt



```
Query: (distance (kocaeli X Y))
Result: ((istanbul 1) (kayseri 5) (edirne 5))

Query: (distance (X edirne Y))
Result: ((izmir 3) (kocaeli 5))

Query: (distance (X bayburt Y))
Result: ((gümüshane 4))

Query: (distance (X Y 5))
Result: ((kocaeli kayseri) (kocaeli edirne))

Query: (distance (X Y Z))
Result:
((kocaeli istanbul 1) (kocaeli kayseri 5) (istanbul kayseri 2) (izmir edirne 3)
 (kocaeli edirne 5) (gümüshane bayburt 4))

Query: (route (kocaeli kars))
Result: T

Query: (route (malatya adana))
Result: ()

Query: (route (izmir adana))
Result: ()
```

Figure 6: output.txt